REMARKS

Claims 2-3, 5-21, 22-24, and 26-37 remain pending. Applicants have amended claims 2-3, 5-21 and 22-32 to clarify that the demand loadable components and objects of the present invention are operating system components.

Summary of the Office Action:

The Office Action rejects claims 2-21 and 23-37 under 35 U.S.C. § 102(e) as anticipated by Koppolu, U.S. Patent No. 6,401,099.

Discussion of the Rejections Under 35 U.S.C. § 102(e):

The Office Action rejects claims 2-21 and 23-37 as anticipated by Koppolu. Applicants respectfully submit that Koppolu does not disclose each and every element recited in the pending claims. Accordingly, Applicants request that the rejection be withdrawn.

An aspect of Applicants' invention, as generally illustrated in FIG. 1B and the accompanying description on pages 21-36 of Applicants' specification, is the ability to provide a componentized system architecture. Certain operating system components, such as device drivers 366, a virtual memory manager 362, interprocess communication manager 364, etc. are initially located outside of working memory and are not included as part of the overall operating system. When the services of a particular operating system component are required by, for example, an application program, that particular operating system component is loaded into working memory and registered in a Namespace. Conversely, operating system components that have been loaded into working memory and that are no longer in demand by application programs can be unloaded from memory.

In accordance with the foregoing, Applicants have amended independent claim 3 to recite "a dynamically configurable operating system" that comprises "demand-loadable operating system components" and a "Namespace ... which provides access to one of said operating system components ... as they become needed by applications ... said Namespace managing demand-loading and unloading of said operating system components in said working memory." Thus, claim 3 recites that operating systems components be demand loadable, i.e. loadable when they become needed by applications. Koppolu discloses loading and unloading objects such as document objects (e.g. objects that contain document data such a text document, spreadsheet, drawing etc.). (Koppolu, col. 9, ll. 17-34). However, Koppolu does not disclose a dynamically configurable operating system or demand loadable *operating system components* as claimed.

Thus, Koppolu does not anticipate independent claim 3. Claims 2 and 5-8 depend from claim 3 and are patentable for at least the same reasons.

Applicants have amended independent claim 9 to recite "A method of operating a computer ... wherein ... objects may be loaded during run time ... said application calling a demand-loadable object by causing the name of said object to be presented to said Namespace, wherein said object is an operating system component." As amended, claim 9 includes the element of demand-loadable operating system components, i.e. loadable during run time, when called by an application. As previously discussed, Koppolu does not disclose demand loadable *operating system components*. Thus, Koppolu does not anticipate claim 9. Claims 10-12 depend from claim 9 and are patentable for at least the same the reasons.

Applicants have amended independent claim 13 to recite "at least one object initially stored in said nonworking memory, said object comprising an operating system component ... said application being programmed to cause said one object to be identified to said Namespace whenever said application finds a need for said object during the running of the application ... said Namespace being programmed to: ... causing said one object to be loaded from said storage memory to said working memory." Thus, claim 13 recites a Namespace that can load an object, i.e. *operating system component*, into working memory when the object is identified by an application program. As previously discussed, Koppolu does not disclose demand loading of operating system components as the result of identification by an application program. Thus, Koppolu does not anticipate claim 13. Claims 14-17 depend from claim 13 and are patentable for at least the same reasons.

Applicants have amended independent claim 18 to recite "an application in said working memory, said application needing access to said one object at a particular time during the running of said application, said object comprising an operating system component, and said application being programmed to cause a request for said one object to issue contemporaneously with said particular time ... a Namespace which is programmed to respond to a request from said application for said one object by loading said one object from said storage memory into said working memory." Thus, claim 18 recites that the Namespace loads an object (operating system component) when access to the object is required by an application. Koppolu does not disclose loading operating system components when required by an application program. For at least this reason, Koppolu does not anticipate claim 18.

Claims 19-21 depend from claim 18 and are patentable for the same reasons. Applicants additionally note that claim 19 recites that the "application" notifies the Namespace when an

operating system component is no longer needed and the "object" is "unloaded from said working memory when no longer needed." Claim 20 recites that the "application" notifies the object when it is no longer needed and the "object" is "unloaded from said working memory when no longer needed." Koppolu does not disclose loading and unloading operating system components depending upon application program need. For these additional reasons, claims 19 and 20 are not anticipated by Koppolu.

Applicants have amended independent claim 24 to recite "demand-loadable operating system components... and a Namespace in said working memory which provides in said working memory access to ones of said operating system components as they become needed by applications ... said Namespace managing demand-loading and unloading of said operating system components in said working memory." Koppolu does not disclose demand-loadable operating system components or a Namespace that manages loading and unloading of operating system components. Thus, Koppolu does not anticipate claim 24. Claims 26-32 depend from claim 24 and are patentable for at least the same reasons.

Independent claim 33 recites "A method ... wherein ... objects may be loaded during run time ... a loadable interprocess communication manager resident at link time outside of working memory and dynamically loadable into working memory." The Office Action indicates that Koppolu discloses an interprocess communication manager. However, Koppolu does not disclose a dynamically loadable interprocess communication manger. Accordingly, Koppolu does not anticipate claim 33. Claim 34 depends from claim 33 and is patentable for at least the same reasons.

Independent claim 35 recites "A method ... wherein ... objects may be loaded during run time ... a loadable virtual memory manager resident at link time outside of the memory and dynamically loadable into working memory at run time upon demand of one of the application programs." The Office Action indicates that Koppolu discloses a virtual memory manager. However, Koppolu does not disclose a virtual memory manager that is dynamically loadable, i.e. loaded upon demand of an application program. Thus, Koppolu does not anticipate claim 35. Claims 36-37 depend from claim 35 and are patentable for at least the same reasons.

Discussion of Applicants' Claim of Priority:

The Office Action Summary indicates that "Acknowledgement is made for domestic priority under 35 U.S.C. §§ 120 and/or 121." Applicants priority claim is to provisional patent

OIPE JC100 MAR 14 2003 PATENT & TRADEMARK OFFICE

application 60/099,562. Accordingly, the proper statutory basis for Applicants' priority claim is 35 U.S.C. § 119(e). Applicants request acknowledgment of the priority claim under § 119(e) in the next Office Communication.

Conclusion

The application is considered in good and proper form for allowance, and the Examiner is respectfully requested to pass this application to issue. If, in the opinion of the Examiner, a telephone conference would expedite the prosecution of the subject application, the Examiner is invited to call the undersigned attorney.

Respectfully submitted,

Mark Joy, Reg. No. 35,562
LEYDIG, VOIT & MAYER, LTD.
Two Prudential Plaza, Suite 4900
180 North Stetson
Chicago, Illinois 60601-6780
(312) 616-5600 (telephone)
(312) 616-5700 (facsimile)

Date: March 10, 2003

OIPE JC100 MAR 14 2003 PATENT & TRADEMARK OFFICE

CERTIFICATE OF MAILING

I hereby certify that this RESPONSE TO OFFICE ACTION (along with any documents referred to as being attached or enclosed) is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, Washington, D.C. 20231.

Date: 3-10-03 _____ _____

OIPE JC100 MAR 14 2003 PATENT & TRADEMARK OFFICE

PATENT
Attorney Docket No. 216549

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Forin et al.

Art Unit: 2151

Application No. 09/282,238

Examiner: The T. Ho

Filed: March 31, 1999

For: A HIGHLY COMPONENTIZED SYSTEM ARCHITECTURE WITH DEMAND-LOADING NAMESPACE AND PROGRAMMING MODEL

AMENDMENTS TO CLAIMS
MADE IN RESPONSE TO OFFICE ACTION DATED DECEMBER 9, 2002

IN THE CLAIMS:

Amend claims 2-3, 5-7, 9, 13, 18, 24, 26-28, as follows:

2. (Twice Amended). The method of Claim 3 wherein said demand-loadable operating system components are initially provided in one of:

(a) a memory within said computer;

(b) a location external of said computer.

3. (Twice Amended). A method of providing a dynamically configurable operating system [software executable] on a computer having a working memory, comprising:

providing demand-loadable operating system components initially stored outside of said working memory, each operating system component having an entry point comprising a constructor for an object, and

providing a Namespace in said working memory which provides in said working memory access to ones of said operating system components as they become needed by applications running in said computer, said Namespace managing demand-loading and unloading of said operating system components in said working memory.

5. (Amended). The method of Claim 3 further comprising providing applications in said working memory which rely on said Namespace to furnish access to ones of said operating system components in said working memory as they become needed by ones of said applications.

6. (Amended). The method of Claim 5 wherein each operating system component comprises an object, and wherein the step of providing each demand-loadable component comprises:

providing an IUnknown interface in the object having the following methods:

(a) add reference for incrementing a count of the number of applications requiring the object;

(b) release reference for decrementing a count of the number of applications requiring the object; wherein said Namespace is responsive to said count in said managing of said demand-loading and unloading.

7. (Amended). The method of Claim 5 wherein each operating system component comprises an object, and wherein the step of providing a demand-loadable operating system component comprises:

providing an IUnknown interface in the object having a QueryInterface method of providing access to the methods of the object to an application invoking QueryInterface.

9. (Amended). A method of operating a computer having a working memory wherein applications and objects may be loaded during run time, comprising:

providing a Namespace in said computer;

running an application in said computer;

said application calling a demand-loadable object by causing the name of said object to be presented to said Namespace, wherein said object is an operating system component;

in response to being presented with the name of said object, said Namespace returning to said application an IUnknown pointer of said object;

upon return of said IUnknown pointer of said object, said application using said IUknown pointer to call a QueryInterface method of said object and request a pointer to a desired interface;

said QueryInterface method returning said desired interface, whereby said application can invoke a desired method through said interface.

13. (Amended). A computer having a working memory and access to a storage memory, said computer comprising:

an application capable of being loaded in to said working memory and running in said computer;

at least one object initially stored in said nonworking memory, said object comprising an operating system component;

a Namespace in said working memory;

said application being programmed to cause said one object to be identified to said Namespace whenever said application finds a need for said object during the running of said application;

said Namespace being programmed to:

(a) respond to said application identifying said one object by determining whether said one object is currently registered in Namespace, and if it is not registered, then,

(b) causing said one object to be loaded from said storage memory to said working memory and,

(c) registering said one object in said Namespace,

(d) upon said object being registered in said Namespace, returning to said application a pointer to said object.

18. (Amended). A computer having a working memory and access to a storage memory, said storage memory holding at least one object, said computer comprising:

an application in said working memory, said application needing [to] access to said one object at a particular time during the running of said application, said object comprising an operating system component, and said application being programmed to cause a request for said one object to issue contemporaneously with said particular time;

a Namespace which is programmed to respond to a request from said application for said one object by loading said one object from said storage memory into said working memory and then providing said application with a pointer to said one object.

24. (Twice Amended). A system residing in a working memory of a computer and in a storage memory, said system comprising:

demand-loadable operating system components initially stored in said storage memory, each operating system component having an entry point comprising a constructor for an object, and

a Namespace in said working memory which provides in said working memory access to ones of said operating system components as they become needed by applications running in said computer, said Namespace managing demand-loading and unloading of said operating system components in said working memory.

26. (Amended). The system of Claim 24 further comprising applications in said working memory which rely on said Namespace to furnish access to ones of said operating system components in said working memory as they become needed by ones of said applications.

27. (Amended). The system of Claim 26 wherein each component operating system comprises an object, and [wherein each demand-loadable component comprises] further comprising:

an IUnknown interface in the object having the following methods:

(a) add reference for incrementing a count of the number of applications requiring the object;

(b) release reference for decrementing a count of the number of applications requiring the object; wherein said Namespace is responsive to said count in said managing of said demand-loading and unloading.

28. (Amended). The system of Claim 27 [wherein each component comprises an object, and wherein each demand-loadable component comprises] further comprising:

an IUnknown interface in the object having a QueryInterface method of providing access to the methods of the object to an application invoking QueryInterface.

Cancel claim 4, 25.

OIPE JC100 MAR 14 2003 PATENT & TRADEMARK OFFICE

PATENT
Attorney Docket No. 216549

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Forin et al.

Art Unit: 2151

Application No. 09/282,238

Examiner: The T. Ho

Filed: March 31, 1999

For: A HIGHLY COMPONENTIZED SYSTEM ARCHITECTURE WITH DEMAND-LOADING NAMESPACE AND PROGRAMMING MODEL

PENDING CLAIMS AFTER AMENDMENTS MADE IN RESPONSE TO OFFICE ACTION DATED DECEMBER 9, 2002

What is claimed is:

2. The method of Claim 3 wherein said demand-loadable operating system components are initially provided in one of:

(a) a memory within said computer;

(b) a location external of said computer.

3. A method of providing a dynamically configurable operating system on a computer having a working memory, comprising:

providing demand-loadable operating system components initially stored outside of said working memory, each operating system component having an entry point comprising a constructor for an object, and

providing a Namespace in said working memory which provides in said working memory access to ones of said operating system components as they become needed by applications running in said computer, said Namespace managing demand-loading and unloading of said operating system components in said working memory.

5. The method of Claim 3 further comprising providing applications in said

working memory which rely on said Namespace to furnish access to ones of said operating system components in said working memory as they become needed by ones of said applications.

6. The method of Claim 5 wherein each operating system component comprises an object, and wherein the step of providing each demand-loadable component comprises:

providing an IUnknown interface in the object having the following methods:

(a) add reference for incrementing a count of the number of applications requiring the object;

(b) release reference for decrementing a count of the number of applications requiring the object; wherein said Namespace is responsive to said count in said managing of said demand-loading and unloading.

7. The method of Claim 5 wherein each operating system component comprises an object, and wherein the step of providing a demand-loadable operating system component comprises:

providing an IUnknown interface in the object having a QueryInterface method of providing access to the methods of the object to an application invoking QueryInterface.

8. The method of Claim 5 wherein said object is a COM object.

9. A method of operating a computer having a working memory wherein applications and objects may be loaded during run time, comprising:

providing a Namespace in said computer;

running an application in said computer;

said application calling a demand-loadable object by causing the name of said object to be presented to said Namespace, wherein said object is an operating system component;

in response to being presented with the name of said object, said Namespace returning to said application an IUnknown pointer of said object;

upon return of said IUnknown pointer of said object, said application using said IUknown pointer to call a QueryInterface method of said object and request a pointer to a desired interface;

said QueryInterface method returning said desired interface, whereby said application can invoke a desired method through said interface.

10. The method of Claim 9, wherein said computer comprises a loader, and wherein said Namespace, responds to being presented with the name of said object in that said Namespace:

determines whether the name of said object is currently registered in said Namespace, and, if so, carries out the step of returning said pointer;

if said name is not currently registered, causes said loader to load said object into said working memory and registers said name in said Namespace, and then carries out the step of returning said pointer.

11. The method of Claim 10 wherein said object has a constructor and an entry point, and wherein the loading of said object comprises:

said loader invoking said constructor;

said constructor finding said entry point of said object and calling an executable at said entry point;

said executable causing space in said working memory to be allocated for a VTable, an Interface and an Implementation of said object and producing a pointer to said memory space, said pointer comprising said IUnknown pointer.

12. The method of Claim 11 further comprising:

loading said VTable, Interface and Implementation in the space in said working memory allocated therefor;

initializing the state of said object including said VTable and interface pointers.

13. A computer having a working memory and access to a storage memory, said computer comprising:

an application capable of being loaded in to said working memory and running in said computer;

at least one object initially stored in said nonworking memory, said object comprising an operating system component;

a Namespace in said working memory;

said application being programmed to cause said one object to be identified to said Namespace whenever said application finds a need for said object during the running of said application;

said Namespace being programmed to:

(a) respond to said application identifying said one object by

determining whether said one object is currently registered in Namespace, and if it is not registered, then,

(b) causing said one object to be loaded from said storage memory to said working memory and,

(c) registering said one object in said Namespace,

(d) upon said object being registered in said Namespace, returning to said application a pointer to said object.

14. The computer of Claim 13 wherein said object comprises a VTable, plural interfaces and corresponding methods thereof, and plural implementations, one of said interfaces comprising an IUknown interface including a QueryInterface method, and wherein said application finds a need for said one object because said application needs a particular one said interfaces of said one object, wherein: said pointer to said object returned by said Namespace comprises an IUnknown pointer to said IUknown interface of said one object;

said application is programmed to use said IUnknown pointer to access said IUknown interface of said one object and to invoke the QueryInterface method thereof to access said particular one interface.

15. The computer of Claim 14 wherein said computer further comprises a loader in said working memory capable of loading objects from said storage memory to said working memory, said Namespace causing said one object to be loaded by causing said loader to load said one object.

16. The computer of Claim 15 wherein said object has a constructor and an entry point, and wherein said loader is programmed such that said loader:

invokes said constructor;

said constructor is programmed to find said entry point of said object and call an executable at said entry point;

said executable is programmed to cause space in said working memory to be allocated for a VTable, an Interface and an Implementation of said object and producing a pointer to said memory space, said pointer comprising said IUnknown pointer.

17. The computer of Claim 16 wherein:

said loader is further programmed to load said VTable, Interface and Implementation in the space in said working memory allocated therefor;

said loader is programmed to initialize the state of said object including said

VTable and interface pointers.

18. A computer having a working memory and access to a storage memory, said storage memory holding at least one object, said computer comprising:

an application in said working memory, said application needing access to said one object at a particular time during the running of said application, said object comprising an operating system component, and said application being programmed to cause a request for said one object to issue contemporaneously with said particular time;

a Namespace which is programmed to respond to a request from said application for said one object by loading said one object from said storage memory into said working memory and then providing said application with a pointer to said one object.

19. The computer of Claim 18 wherein:

said application is programmed to notify said Namespace whenever it no longer needs access to said one object;

said Namespace is further programmed to permit said one object to be unloaded from said working memory when no longer needed.

20. The computer of Claim 18 wherein:

said application is programmed to notify said one object it no longer needs access to it, whereby said object notifies said NameSpace said object is no longer needed;

said Namespace is further programmed to permit said one object to be unloaded from said working memory when no longer needed.

21. The computer of Claim 18 wherein said Namespace permits said one object to remain in working memory after being no longer needed by said application in order to permit other applications to access said one object.

23. The system of Claim 24 wherein said storage memory comprises one of:

(a) a memory within said computer;

(b) a memory external of said computer.

(c) the output of another software component such as a compiler.

24. A system residing in a working memory of a computer and in a storage memory, said system comprising:

demand-loadable operating system components initially stored in said storage

memory, each operating system component having an entry point comprising a constructor for an object, and

a Namespace in said working memory which provides in said working memory access to ones of said operating system components as they become needed by applications running in said computer, said Namespace managing demand-loading and unloading of said operating system components in said working memory.

26. The system of Claim 24 further comprising applications in said working memory which rely on said Namespace to furnish access to ones of said operating system components in said working memory as they become needed by ones of said applications.

27. The system of Claim 26 wherein each component operating system comprises an object, and further comprising:

an IUnknown interface in the object having the following methods:

(a) add reference for incrementing a count of the number of applications requiring the object;

(b) release reference for decrementing a count of the number of applications requiring the object; wherein said Namespace is responsive to said count in said managing of said demand-loading and unloading.

28. The system of Claim 27 further comprising:

an IUnknown interface in the object having a QueryInterface method of providing access to the methods of the object to an application invoking QueryInterface.

29. The system of Claim 26 wherein said object is a COM object.

30. The system of Claim 27 further comprising a loader in said working memory, and wherein said Namespace is responsive to being presented with the name of one of said objects by:

returning a pointer to said object if said object is registered in said Namespace and returning to said application a pointer to said object;

if said name is not currently registered, causing said loader to load said object into said working memory.

31. The system of Claim 30 wherein said object has a constructor invoked by said loader, an entry point and an working memory space-allocating executable called by said

constructor at said entry point.

32. The system of Claim 31 wherein said object comprises a VTable, an Interface and an Implementation in locations in said working memory allocated by said executable.

33. A method of operating a computer having a working memory wherein applications and objects may be loaded during run time, comprising:

providing a Namespace in the computer;

providing a kernel resident in the working memory at run time;

providing a loadable interprocess communication manager resident at link time outside of the working memory and dynamically loadable into the working memory at run time, the interprocess communication manager having an IUnknown pointer, a QueryInterface method and plural interfaces;

running an application in the computer;

the application presents to the Namespace the name of the interprocess communication manager;

in response to the Namespace being presented by the application with the name of the interprocess communication manager, the Namespace returning to the application the IUnknown pointer of the interprocess communication manager;

upon return of the IUnknown pointer, the application using the IUknown pointer to call the QueryInterface method of the interprocess communication manager, and requesting through the QueryInterface method a pointer to a desired interface of the interprocess communication manager;

the QueryInterface method returning the desired interface, whereby the application can invoke a desired method through the interface.

34. The method of Claim 33, wherein the computer comprises a loader in the working memory, and wherein the Namespace, prior to providing the IUnknown pointer, performs the steps of:

determining whether the interprocess communication manager is currently registered in the Namespace;

if the interprocess communication manager is not currently registered, causing the loader to load the interprocess communication manager into the working memory and registering the interprocess communication manager in the Namespace.

35. A method of operating a computer having a working memory wherein

applications and objects may be loaded during run time, the method comprising:

providing a Namespace in the computer;

providing a kernel resident in the working memory at run time;

providing a loadable virtual memory manager resident at link time outside of the memory and dynamically loadable into the working memory at run time upon demand of one of the application programs, the virtual memory manager having an IUknown pointer, a QueryInterface method and plural interfaces;

running an application in said computer;

said application presenting the name of the virtual memory manager to the Namespace;

in response to the Namespace being presented by the application with the name of the virtual memory manager, the Namespace returning to the application the IUknown pointer of the virtual memory manager;

upon return of the IUknown pointer, the application using the IUknown pointer to call the QueryInterface method of the virtual memory manager and to request through the QueryInterface method a pointer to a desired one of the plural interfaces of the virtual memory manager;

the QueryInterface method returning the desired interface, whereby the application can invoke a desired method through the Interface.

36. The method of Claim 35, wherein the computer comprises a loader in the working memory, and wherein the Namespace, prior to providing the IUknown pointer, performs the steps of:

determining whether the name of the virtual memory manager is currently registered in the Namespace;

if the name is not currently registered, causing the loader to load the virtual memory manager into the working memory and registering the virtual memory manager in the Namespace.

37. The method of Claim 35 wherein the plural interfaces of the virtual memory manager comprise:

virtual memory space interface (VMSpace);

virtual memory map interface (VMMap);

virtual memory view interface (VMView).